VolleyBlog

Table of Contents

About

The idea of the project is to create a small, self-hosted internet blog about volleyball.

Objective

The objective of this project is to create a self hosted volleyball oriented blog website. Journey through the whole web development process will give the students the chance to experience both frontend and backend development.

Features

Users:

- Create an account
- Update their data
- Delete their account

Blogs:

- Create a new blog post
- Update the post data
- Comment on the post
- Like the post
- Remove the post

Models

User

The user model will be composed of the following fields:

- first_name
- last_name
- date_of_birth
- username
- posts
- volleyball_team

Definitions The first_name, last_name, date_of_birth are pretty clear so we are going to skip them.

1. Username

   The username field will be of type string. The value of which has to be unique among all of the usernames in the blog database - there can be no two same usernames.

2. Posts

   The posts array will be another field in the user document in the database. It will contain all of the IDs of the posts the user has published. This not only will allow for easier posts removal in case of account deletion, but also make is easier to, for example display them on the user profile.

3. Volleyball Team

   The volleyball team will be an optional field, if the user wants to share his current club, they are able to.

Creating the account
The user is presented with a form asking him for his details:

- first_name*
- last_name*
- date_of_birth*
- username*
- volleyball_team

After successful registration a new document is created in the database with the user data, the user_id is saved to the local / session storage and the user is redirected to the Home page.

Updating the user data
The user can at any time access his profile details and change the following:

- username (to another unique)
- first_name
- last_name
- volleyball_team

Deleting the account
The user is allowed to delete his account at any time, in result of which all of his posts are removed, then the account is deleted. The deletion will be invoked by an option in the user profile page.

Blog

The blog model will be composed with the following:

- title*
- headline*
- author*
- date_of_creation*
- image_prompt*
- image
- comments
- likes

Definitions

1. Title

   The title of the post, the string value that is used to search for a post and the that is displayed first.

2. Headline

   The headline will be a string value displayed right below the title. It's a brief introduction to what the rest of the publication is about.

3. Author

   The author field is automatically assigned with the string value saved in the storage as user_id.

4. Date_of_creation

   The date_of_creation field is also automatically assigned with the string value of the current time in the following format (dd/mm/yyyy).

5. Image_prompt

   The image_prompt string field is optional. The value given in here will serve as a prompt for the Dalle-3 image generator. If no value is given, the headline will be used as the prompt.

6. Image

   The image string field will not be filled by the user. The value of this field will be assigned to the url / file_name of the Dalle-3 product image.

7. Comments

   The comments List[Dict] field is asssigned an empty list at post creation. Later on, when a user comments on the post, a new entry is added in the format of a JSON dictionary in the following format:

   - {"author": author_id, "comment": comment_body, "date": date}

8. Likes

   The likes List[str] field will be assigned an empty array. Then when a user likes / dislikes the post, his ID will be added / removed from the list. This way we can not only access of the users that have liked the post, but also get the number of likes.

Creating a new blog post
The user is presented with a form asking for the following:

1. title
2. headline
3. image_prompt

Then, depending on whether the user has or has not filled in the image_prompt field, either its' or the headline's field value is sent as the prompt to the Dalle-3 image generator through an API. After receiving the response the image is saved as **./assets/posts/<post_id>/image.jpg**.

Then a new document is created in the database with fields initialised as follows:

1. title - title field value
2. headline - headline field value
3. image_prompt - image_prompt value if not empty else None
4. image - the image path
5. date_of_creation - current date
6. comments - []
7. likes - []

After the successful post creation, its' ID is added to the user's posts array.

Finally, the user is redirected to the dedicated post page.

Updating the post data
The user is presented again with the forms, this time filled with the values read from the database. After successful changes followed by the push of the save button the data is changed in the database and the user is redirected to the post page.

Removing the post data
The removal process is invoked by the push of the delete icon in the right-bottom corner of the blog post. The user is asked for confirmation, after which the post is deleted from the databse and it's ID is removed from the user's posts array.

Commenting the post
Any user that has signed is able to comment on the post. The comments will not be nested (no comments on another comment).

Liking the post
Similarly, any post can be liked. Whenever a user likes a post, his ID is added to the likes array of the post.

Grading information

Requirements

1. Proper file naming
2. Snake case naming convention
3. There can be no UI elements without proper implementation (no buttons without effect)

Workspace preparation

First launch

Creating the virtual environment

```
python3 -m venv venv
```

Activating the virtual environment

```
./venv/venv/activate
```

Installing the requirements

```
pip3 install -r requirements.txt
```

Then begin working on the project.

Relaunching

Activating the virtual environment

```
./venv/venv/activate
```

Contributions

This project was prepard entirely by Jakub Nenczak
Github link